Data Analysis on DmartReady Online Store

Introduction

The DmartReady Online Store Analysis is a comprehensive data-driven project aimed at understanding the operational, financial, and customer behavioral patterns of DmartReady, an online retail platform. The analysis leverages SQL and Python to extract, transform, and analyze large datasets, providing valuable insights into sales trends, customer demographics, product performance, and operational efficiency. By examining key metrics such as total sales, order volumes, customer engagement, and marketing performance, the project identifies factors influencing the business's growth and profitability.

Objectives

The primary objectives of this analysis include: Understanding Sales Performance: Analyze total and average sales, sales by category, and revenue contribution by gender and marketing campaigns. Customer Behavior Analysis: Study customer demographics, average time spent on the website, and buying patterns by state, city, and age group. Product Analysis: Identify top-performing products, products with the highest discount rates, and products with the highest customer engagement. Operational Efficiency: Evaluate delivery times, ship modes, and cancellation rates across states and cities to improve logistics. Marketing Insights: Assess the effectiveness of different marketing platforms in driving sales and revenue. Profitability Assessment: Determine profit margins across various product categories.

Data Analysis Workflow: Importing necessary libraries

.

In [2]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import mysql.connector
import plotly.graph_objects as go
import plotly.express as px
```

Data Collection: Connect with SQL database, Extract data using SQL queries.

In [204…
```python
db = mysql.connector.connect(
    host='127.0.0.1',
    user='root',
    password='1234',
    database='dmart'
)
cursor = db.cursor() # Creates a cursor object from the established connection
```

In [266…

```
query = """ SELECT * from dmart.dmartready"""
cursor.execute(query)  # Executes a SELECT query
data = cursor.fetchall() # Fetches all rows from the executed query
df = pd.DataFrame(data, columns = ["CustomerID", "ProductID", "OrderID", "Age",
                                   "DiscountPrice", "Category", "State", "City",
                                   "Rating", "Marketing/Advertisement", "ShipMod
                                   "PaymentMethod", "PinCode", "TotalOrderValue"
                                   "No.of_Clicks", "Year", "Month", "ShippingCha
df.head()
# Converts the fetched SQL data into a pandas DataFrame for further analysis.
```

Out[266…

	CustomerID	ProductID	OrderID	Age	Gender	ProductName	MRP	DiscountP
0	YSB75	BW653	479577309	28	Female	Ariel Matic Top Load Liquid Detergent	977.44	90
1	FUS93	XV061	634865221	47	Male	VSR Channa Dal	834.09	45
2	AJP28	GF695	113166210	63	Female	Tenali Double Horse Chana Dal	1095.20	100
3	URC55	VM478	740539230	41	Male	Tata Tea	748.16	38
4	ZOP23	XD230	156544145	22	Female	VSR Channa Dal	1249.04	91

5 rows × 29 columns

Understanding the data & cleaning:

In [267…

```
# Used to generate descriptive statistics for the numerical columns in a DataFra
df.describe().round()
```

Out[267…

	OrderID	Age	MRP	DiscountPrice	BillNumber	TimeSpentonWebsite
count	25000.0	25000.0	25000.0	25000.0	2.500000e+04	25000.0
mean	498712983.0	44.0	1014.0	736.0	5.016699e+11	10.0
std	288189483.0	15.0	571.0	441.0	2.887199e+11	11.0
min	19128.0	18.0	20.0	11.0	2.349470e+07	1.0
25%	250374576.0	31.0	524.0	368.0	2.540238e+11	3.0
50%	499731127.0	44.0	1014.0	713.0	5.019725e+11	5.0
75%	747102105.0	57.0	1507.0	1061.0	7.519322e+11	13.0
max	999913093.0	70.0	2000.0	1876.0	9.999160e+11	60.0

In [270…
```
# Checking rows & columns present in data frame
df.shape
```

Out[270…
```
(25000, 29)
```

In [272…
```
df.columns
```

Out[272…
```
Index(['CustomerID', 'ProductID', 'OrderID', 'Age', 'Gender', 'ProductName',
       'MRP', 'DiscountPrice', 'Category', 'State', 'City', 'Subscription',
       'BillNumber', 'TimeSpentonWebsite', 'Rating', 'Marketing/Advertisement',
       'ShipMode', 'OrderStatus', 'OrderDate', 'DeliveryDate',
       'CancellationDate', 'PaymentMethod', 'PinCode', 'TotalOrderValue',
       'PaymentStatus', 'No.of_Clicks', 'Year', 'Month', 'ShippingCharges'],
      dtype='object')
```

In [274…
```
# Checking overview of data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 29 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   CustomerID               25000 non-null  object 
 1   ProductID                25000 non-null  object 
 2   OrderID                  25000 non-null  int64  
 3   Age                      25000 non-null  int64  
 4   Gender                   25000 non-null  object 
 5   ProductName              25000 non-null  object 
 6   MRP                      25000 non-null  float64
 7   DiscountPrice            25000 non-null  float64
 8   Category                 25000 non-null  object 
 9   State                    25000 non-null  object 
 10  City                     25000 non-null  object 
 11  Subscription             25000 non-null  object 
 12  BillNumber               25000 non-null  float64
 13  TimeSpentonWebsite       25000 non-null  float64
 14  Rating                   25000 non-null  float64
 15  Marketing/Advertisement  25000 non-null  object 
 16  ShipMode                 25000 non-null  object 
 17  OrderStatus              25000 non-null  object 
 18  OrderDate                25000 non-null  object 
 19  DeliveryDate             25000 non-null  object 
 20  CancellationDate         1784 non-null   object 
 21  PaymentMethod            25000 non-null  object 
 22  PinCode                  25000 non-null  int64  
 23  TotalOrderValue          25000 non-null  object 
 24  PaymentStatus            25000 non-null  object 
 25  No.of_Clicks             25000 non-null  int64  
 26  Year                     25000 non-null  int64  
 27  Month                    25000 non-null  object 
 28  ShippingCharges          25000 non-null  object 
dtypes: float64(5), int64(5), object(19)
memory usage: 5.5+ MB
```

In [276…
```
# Using Conditional statement checking for duplicate value in dataframe
if df.duplicated().sum() > 0:
    print('Duplicates are present in dataframe')
```

```
else:
    print('Duplicate does not exist')
```

```
Duplicate does not exist
```

In [278…

```
# Another method of check dupicates
df.duplicated().sum()
```

Out[278…

```
0
```

In [406…

```
df.isna()
```

Out[406…

	CustomerID	ProductID	OrderID	Age	Gender	ProductName	MRP	DiscountP
0	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	
24995	False	False	False	False	False	False	False	F
24996	False	False	False	False	False	False	False	F
24997	False	False	False	False	False	False	False	F
24998	False	False	False	False	False	False	False	F
24999	False	False	False	False	False	False	False	F

25000 rows × 31 columns

In [280…

```
# Check for null values
df.isnull().sum()
```

```
Out[280…  CustomerID                      0
          ProductID                       0
          OrderID                         0
          Age                             0
          Gender                          0
          ProductName                     0
          MRP                             0
          DiscountPrice                   0
          Category                        0
          State                           0
          City                            0
          Subscription                    0
          BillNumber                      0
          TimeSpentonWebsite              0
          Rating                          0
          Marketing/Advertisement         0
          ShipMode                        0
          OrderStatus                     0
          OrderDate                       0
          DeliveryDate                    0
          CancellationDate            23216
          PaymentMethod                   0
          PinCode                         0
          TotalOrderValue                 0
          PaymentStatus                   0
          No.of_Clicks                    0
          Year                            0
          Month                           0
          ShippingCharges                 0
          dtype: int64
```

```
In [20]:  df.rename(columns={
              'TotalOrderValue': 'Sales'}, inplace=True)
```

```
In [26]:  df['Sales'] = df['Sales'].str.replace('abc|xyz', '', regex=True)
```

```
In [30]:  df['Sales'] = df['Sales'].str.strip()  # Removes leading and trailing spaces
```

```
In [32]:  df['Sales'] = df['Sales'].str.replace(r'[!#@]', '', regex=True) # Remove specifi
```

```
In [46]:  # Replace "Cancelled" and "Returned" in the 'Order_Status' column with 0
          df['Sales'] = df['Sales'].where(~df['Sales'].isin(['Cancelled', 'Returned']), 0)
```

```
In [52]:  # Change data type from string to integer
          df['Sales'] = df['Sales'].astype(float)
```

Exploratory Data Analysis & Visualization

```
In [200…  # Total Sales ?
          query = """ select round(sum(total_order_value)) from dmartready"""
          cursor.execute(query)
          data1a = cursor.fetchall()
          data1a
```

```
Out[200…  [(16954109.0,)]
```

In [410…
```
df['TotalOrderValue'].sum().round()
```

Out[410…
```
16954109.0
```

In [184…
```
# Find sales contribution by gender?
# Customer Segmentation
df['AgeRange'] = pd.cut(df['Age'], bins=[18, 25, 35, 45], labels=["18-25", "26-3
customer_segmentation = df.groupby(['Gender', 'AgeRange'])['Sales'].sum()
customer_segmentation
```

Out[184…
```
Gender  AgeRange
Female  18-25        958540.39
        26-35       1362935.26
        36-45       1412049.54
Male    18-25       1272582.93
        26-35       1867735.05
        36-45       1870226.81
Name: Sales, dtype: float64
```

In [194…
```
data = {
    'Gender': ['Female', 'Female', 'Female', 'Male', 'Male', 'Male'],
    'AgeRange': ['18-25', '26-35', '36-45', '18-25', '26-35', '36-45'],
    'Sales': [958540.39, 1362935.26, 1412049.54, 1272582.93, 1867735.05, 1870226
}
df = pd.DataFrame(data)
fig = px.sunburst(
    df,
    path=['Gender', 'AgeRange'],
    values='Sales',
    color='Gender',
    template="plotly_dark",
    title="Customer Segmentation by Gender and Age Range"
)
fig.show()
```

```
In [62]: UniqueOrders = df['OrderID'].nunique()
         print(UniqueOrders)
```

```
25000
```

```
In [54]: # what is the average sales ?
         AverageSalesPerOrder = df.groupby('OrderID')['Sales'].sum().mean()
         AverageSalesPerOrder = round(AverageSalesPerOrder, 2)

         # Output the result with formatting
         print(f"Average Sales per Order: ₹{AverageSalesPerOrder:,.2f}")
```

```
Average Sales per Order: ₹678.16
```

```
In [56]: # Get the total number of products sold in each category.
         query = """ SELECT Category, COUNT(Product_ID) AS TotalProductsSold
         FROM dmartready GROUP BY Category"""
         cursor.execute(query)
         data1 = cursor.fetchall()
         df1 = pd.DataFrame(data1, columns = ["Category", "TotalProductSold"])
         df1.head()
```

Out[56]:

	Category	TotalProductSold
0	Imported	5034
1	Branded	7564
2	Local	12402

In [550…

```
Category = ['Imported', 'Branded', 'Local']
TotalProductSold = [5034, 7564, 12402]

fig = go.Figure(data=[go.Pie(labels=Category, values=TotalProductSold, hole=0.5)
fig.update_traces(textinfo='percent+label', pull=[0, 0, 0.1])  # Set the layout
# Add title and labels
fig.update_layout(
    title="Pie Chart of Product Categories",
    title_x=0.5,
    template="plotly_dark",
    showlegend=True
)
fig.show()
```

In [126…

```
# Get the number of orders placed in each month for a specific year
query = """SELECT Month, COUNT(Order_ID) AS OrdersInMonth
FROM dmartready
WHERE Year = 2023
GROUP BY Month """
```

```
cursor.execute(query)
data1 = cursor.fetchall()
df1 = pd.DataFrame(data1, columns = ["Month", "OrdersInMonth2023"])
# correct month order
month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', '
# Convert the 'Month' column to a categorical type with the specified order
df1['Month'] = pd.Categorical(df1['Month'], categories=month_order, ordered=True
df_sorted1 = df1.sort_values(by=['Month'])
```

In [124…

```
query = """SELECT Month, COUNT(Order_ID) AS OrdersInMonth
FROM dmartready
WHERE Year = 2022
GROUP BY Month """
cursor.execute(query)
data2 = cursor.fetchall()
df2 = pd.DataFrame(data2, columns = ["Month", "OrdersInMonth2022"])
# correct month order
month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', '
# Convert the 'Month' column to a categorical type with the specified order
df2['Month'] = pd.Categorical(df2['Month'], categories=month_order, ordered=True
df_sorted2 = df2.sort_values(by=['Month'])
```

In [122…

```
query = """SELECT Month, COUNT(Order_ID) AS OrdersInMonth
FROM dmartready
WHERE Year = 2021
GROUP BY Month """
cursor.execute(query)
data3 = cursor.fetchall()
df3 = pd.DataFrame(data3, columns = ["Month", "OrdersInMonth2021"])
# correct month order
month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', '
# Convert the 'Month' column to a categorical type with the specified order
df3['Month'] = pd.Categorical(df3['Month'], categories=month_order, ordered=True
df_sorted3 = df3.sort_values(by=['Month'])
```

In [116…

```
# Merge the DataFrames on the 'Month' column
MergedData = pd.merge(df1, df2, on="Month", how="outer")
MergedData = pd.merge(MergedData, df3, on="Month", how="outer")
MergedData
```

Out[116…

	Month	OrdersInMonth2023	OrdersInMonth2022	OrdersInMonth2021
0	January	678	644	681
1	February	680	651	627
2	March	711	706	715
3	April	665	686	715
4	May	722	709	689
5	June	660	697	655
6	July	687	722	711
7	August	706	765	711
8	September	684	620	658
9	October	723	704	714
10	November	711	688	750
11	December	738	733	684

In [548…

```python
# Create bar chart
fig = px.bar(MergedData, x="Month", y=["OrdersInMonth2021", "OrdersInMonth2022",
             title="Orders Comparison Across Years (2021, 2022, 2023)",
             labels={"value": "Orders", "variable": "Year"},
             template="plotly_dark")
fig.show()
```

In [106…

```
query = """select year, count(order_id) as OrdersInYear
from dmartready
group by year """
cursor.execute(query)
data4 = cursor.fetchall()
df4 = pd.DataFrame(data4, columns = ["Year", "OrdersInYear"])
df4_sort = df4.sort_values(by=['Year'])
df4_sort
```

Out[106…

	Year	OrdersInYear
1	2021	8310
0	2022	8325
2	2023	8365

In [414…

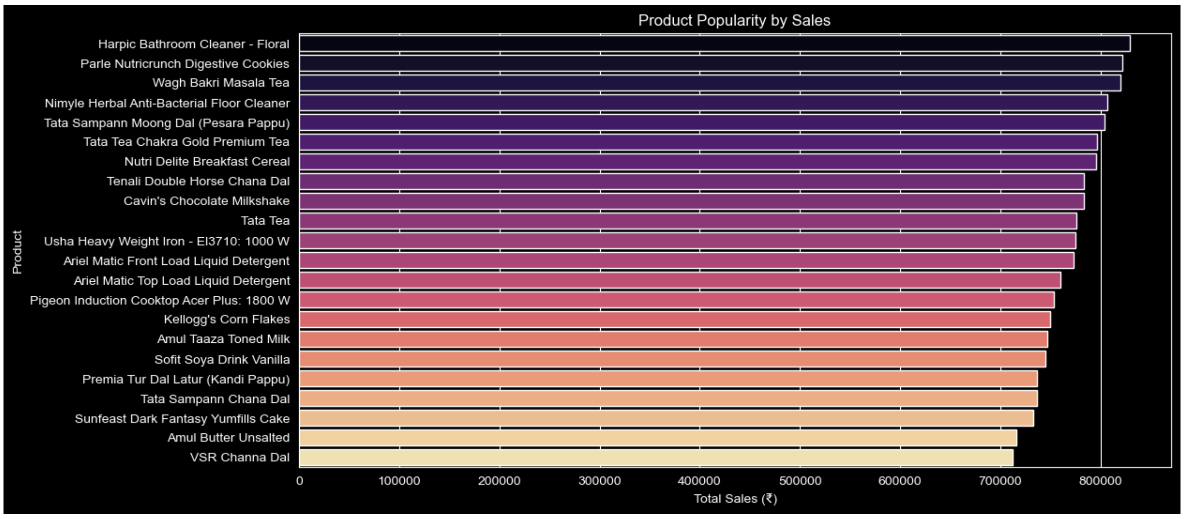
```
# Calculate the total revenue generated from each product
query = """SELECT Product_Name, round(SUM(Total_Order_Value)) AS TotalRevenue
FROM dmartready
GROUP BY Product_Name order by TotalRevenue desc"""
cursor.execute(query)
data5 = cursor.fetchall()
df5 = pd.DataFrame(data5, columns = ["ProductName", "TotalSales"])
df5.head()
```

Out[414…

	ProductName	TotalSales
0	Harpic Bathroom Cleaner - Floral	828936.0
1	Parle Nutricrunch Digestive Cookies	821713.0
2	Wagh Bakri Masala Tea	820168.0
3	Nimyle Herbal Anti-Bacterial Floor Cleaner	807119.0
4	Tata Sampann Moong Dal (Pesara Pappu)	804085.0

In [386…

```
# Product Popularity
plt.style.use('dark_background')
product_popularity = df5.groupby('ProductName')['TotalSales'].sum().sort_values(
plt.figure(figsize=(12, 6))
sns.barplot(x=product_popularity.values, y=product_popularity.index, palette='ma
plt.title('Product Popularity by Sales')
plt.xlabel('Total Sales (₹)')
plt.ylabel('Product')
plt.show()
```



In [418…

```
# Determine the average rating for products in each product_name
query = """SELECT product_name, round(AVG(Rating),2) AS AverageRating
FROM dmartready
GROUP BY product_name order by AverageRating desc"""
cursor.execute(query)
data5a = cursor.fetchall()
df5a = pd.DataFrame(data5a, columns = ["ProductName", "AverageRating"])
df5a.head()
```

Out[418…

	ProductName	AverageRating
0	Ariel Matic Top Load Liquid Detergent	3.05
1	Amul Taaza Toned Milk	3.05
2	Tata Tea Chakra Gold Premium Tea	3.05
3	Tata Tea	3.04
4	Nimyle Herbal Anti-Bacterial Floor Cleaner	3.04

In [25]:
```
# Identify the total sales by each category ?
query = """SELECT Category, round(SUM(Total_Order_Value)) AS TotalRevenue
FROM dmartready
GROUP BY Category order by TotalRevenue desc"""
cursor.execute(query)
data5b = cursor.fetchall()
df5b = pd.DataFrame(data5b, columns = ["Category", "TotalSales"])
df5b.head()
```

Out[25]:

	Category	TotalSales
0	Local	8482035.0
1	Branded	5124795.0
2	Imported	3347279.0

In [73]:
```
# Total Sales by state ?
query = """SELECT State, round(SUM(Total_Order_Value)) AS TotalRevenue
FROM dmartready
GROUP BY State order by TotalRevenue desc"""
cursor.execute(query)
data5c = cursor.fetchall()
df5c = pd.DataFrame(data5c, columns = ["State", "TotalSales"])
df5c
```

Out[73]:

	State	TotalSales
0	Maharashtra	4298237.0
1	Andhra Pradesh	4249018.0
2	Gujarat	4225412.0
3	Telangana	4181442.0

In [366…
```
# Create a Donut chart using Plotly
fig = px.pie(df5c,
             names='State',  # Use the 'State' column for the segments
             values='TotalSales',  # Use 'TotalSales' as the size of each segmen
             title="Total Sales by State",
             template="plotly_dark",
             hole=0.5)  # This creates the donut shape by making a hole in the c

# Show the interactive plot
fig.show()
```

In [128…

```
# Total Sales by top 10 City ?
query = """SELECT City, round(SUM(Total_Order_Value)) AS TotalRevenue
FROM dmartready
GROUP BY City order by TotalRevenue desc"""
cursor.execute(query)
data5d = cursor.fetchall()
df5d = pd.DataFrame(data5d, columns = ["City", "TotalSales"])
df5d.head(10)
```

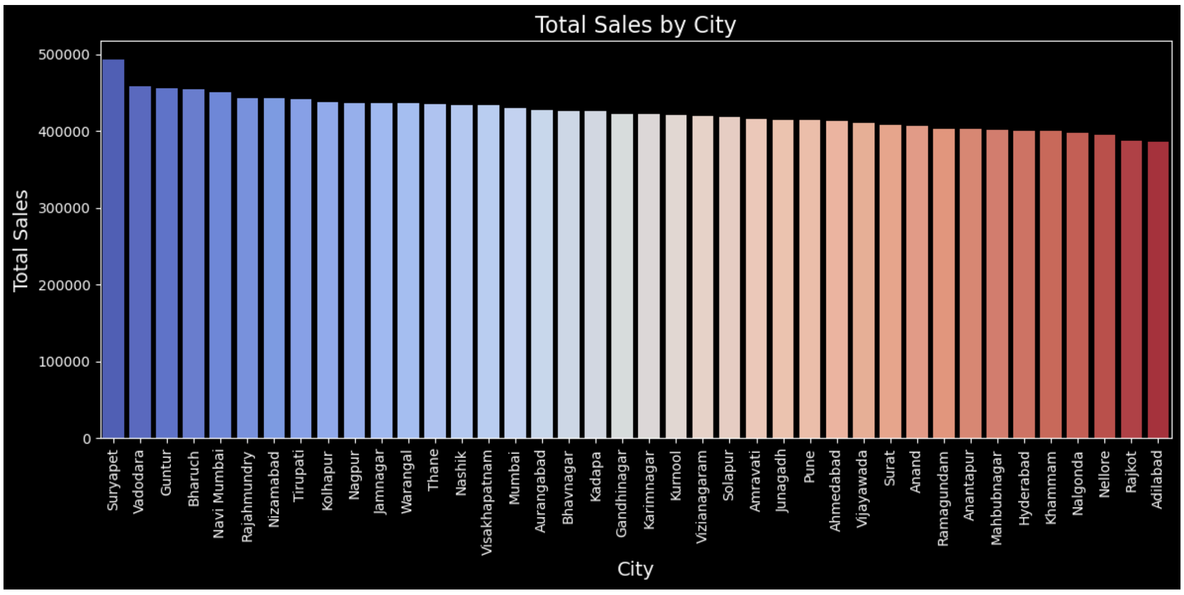
Out[128…

	City	TotalSales
0	Suryapet	492450.0
1	Vadodara	457906.0
2	Guntur	456108.0
3	Bharuch	453977.0
4	Navi Mumbai	450947.0
5	Rajahmundry	443223.0
6	Nizamabad	442528.0
7	Tirupati	441999.0
8	Kolhapur	436959.0
9	Nagpur	436765.0

In [352…

```
# Set the figure size for the plot
plt.figure(figsize=(12, 6))

# Plotting the bar chart
sns.barplot(x='City', y='TotalSales', data=df5d, hue = 'City', palette='coolwarm
plt.xticks(rotation=90)
plt.title('Total Sales by City', fontsize=16)
plt.xlabel('City', fontsize=14)
plt.ylabel('Total Sales', fontsize=14)
plt.tight_layout()  # Adjust the plot to fit everything
plt.show()
```



In [308…

```
# Identify Operational Efficiency for delivery?
# Convert Columns to Datetime
df['OrderDate'] = pd.to_datetime(df['OrderDate'], format='%d-%m-%Y')
df['DeliveryDate'] = pd.to_datetime(df['DeliveryDate'], format='%d-%m-%Y')
# Calculate DeliveryTime
df['DeliveryTime'] = (df['DeliveryDate'] - df['OrderDate']).dt.days
# Check for DeliveryTime column
print(df[['OrderDate', 'DeliveryDate', 'DeliveryTime']])
```

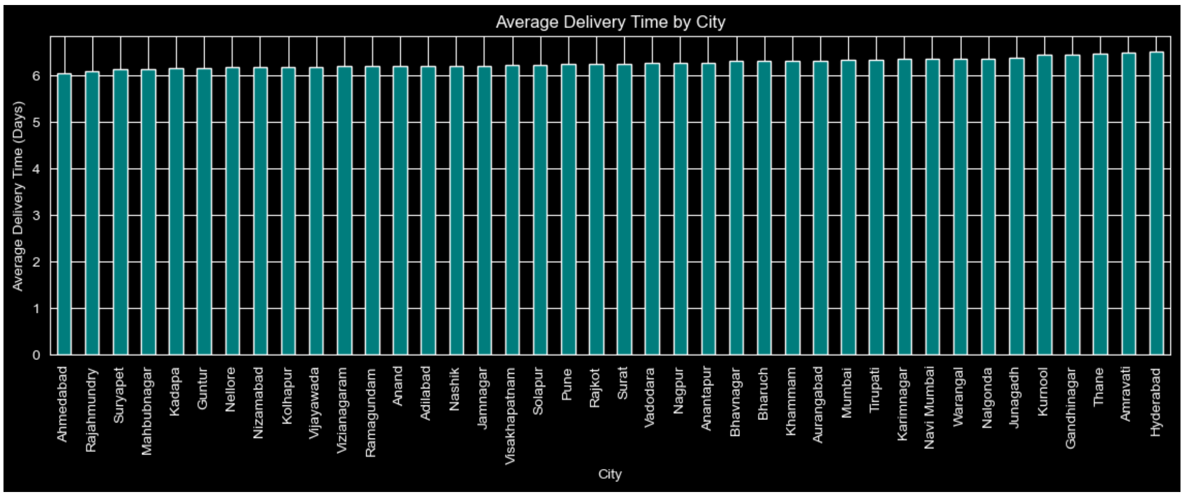
```python
# Calculate average delivery time by city
avg_delivery_time = df.groupby('City')['DeliveryTime'].mean()
```

```
       OrderDate DeliveryDate  DeliveryTime
0     2022-05-05   2022-05-13             8
1     2021-08-13   2021-08-20             7
2     2021-10-17   2021-10-26             9
3     2021-08-25   2021-08-27             2
4     2023-12-08   2023-12-10             2
...          ...          ...           ...
24995 2021-04-02   2021-04-04             2
24996 2021-04-10   2021-04-13             3
24997 2023-09-01   2023-09-05             4
24998 2023-07-13   2023-07-23            10
24999 2022-10-26   2022-11-05            10

[25000 rows x 3 columns]
```

In [430…

```python
# Plotting
plt.style.use('dark_background')
plt.figure(figsize=(14, 4))
avg_delivery_time.sort_values().plot(kind='bar', color='teal')
plt.title('Average Delivery Time by City')
plt.xlabel('City')
plt.ylabel('Average Delivery Time (Days)')
plt.xticks(rotation=90)
plt.show()
```



In [434…

```python
# Total sales by each types of members ?
query = """SELECT Subscription, round(SUM(Total_Order_Value)) AS TotalRevenue
FROM dmartready
GROUP BY Subscription order by TotalRevenue desc"""
cursor.execute(query)
data5e = cursor.fetchall()
df5e = pd.DataFrame(data5e, columns = ["SubscriptionCategory", "TotalSales"])
df5e.head()
```

Out[434…

	SubscriptionCategory	TotalSales
0	Freepass	9720905.0
1	Premium	5173185.0
2	Premium Plus	2060020.0

In [436…

```
fig = px.pie(df5e,
             names='SubscriptionCategory', values='TotalSales', title="Total Sal
             color='SubscriptionCategory',  # Color the slices by the MarketingP
             hover_data=['TotalSales'],  # Show the total sales on hover
             labels={'TotalSales': 'TotalSales', 'SubscriptionCategory': 'Subscr
             template="plotly_dark"  # theme to 'plotly_dark' for a dark backgro
            )
fig.show()
```

In [67]:

```
# Total sales by each marketing campaign ?
query = """SELECT `Marketing/Advertisement`, round(SUM(Total_Order_Value)) AS To
FROM dmartready
GROUP BY `Marketing/Advertisement` order by TotalRevenue desc"""
cursor.execute(query)
data5f = cursor.fetchall()
df5f = pd.DataFrame(data5f, columns = ["MarketingPlatform", "TotalSales"])
df5f.head()
```

Out[67]:

	MarketingPlatform	TotalSales
0	Instagram	5755979.0
1	Facebook	4219599.0
2	Other	2803532.0
3	Friends	2779127.0
4	TV	1395873.0

In [326…

```
fig = px.pie(df5f,
             names='MarketingPlatform', values='TotalSales', title="Total Sales
             color='MarketingPlatform',  # Color the slices by the MarketingPlat
             hover_data=['TotalSales'],  # Show the total sales on hover
             labels={'TotalSales': 'Total Sales', 'MarketingPlatform': 'Marketin
             template="plotly_dark"  # theme to 'plotly_dark' for a dark backgro
            )
fig.show()
```

In [202…

```
# Find the average time spent on the website by customers who made a purchase in
query = """select state, round(avg(time_spent_on_website),2) as AverageTimeSpent
cursor.execute(query)
data6 = cursor.fetchall()
df6 = pd.DataFrame(data6, columns = ["State", "ProductPurchased"])
df6.head()
```

Out[202…

	State	AverageTimeSpent
0	Gujarat	9.99
1	Maharashtra	10.10
2	Telangana	10.26
3	Andhra Pradesh	10.50

In [446…

```
# Find the average time spent on the website by customers who made a purchase in
query = """select city, round(avg(time_spent_on_website),2) as AverageTimeSpent
order by AverageTimeSpent desc"""
cursor.execute(query)
data7 = cursor.fetchall()
df7 = pd.DataFrame(data7, columns = ["City", "AverageTimeSpent"])
df7.head()
```

Out[446…

	City	AverageTimeSpent
0	Ramagundam	11.69
1	Vijayawada	11.50
2	Karimnagar	11.34
3	Junagadh	11.18
4	Rajkot	11.17

In [452…

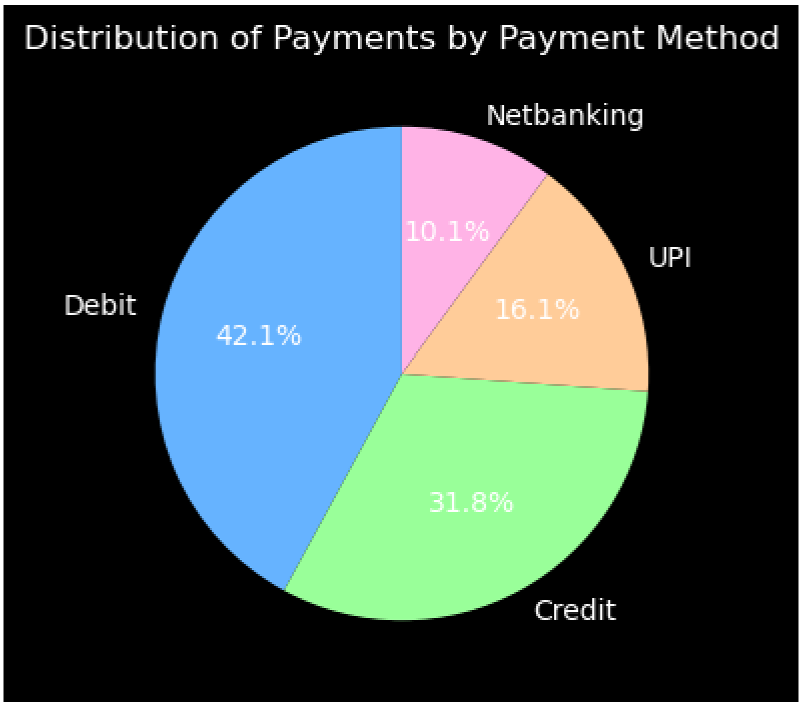
```
# Find the total sales (OrderValue) by payment method.
query = """SELECT Payment_Method, round(SUM(Total_Order_Value)) AS TotalSales
FROM dmartready
GROUP BY Payment_Method order by TotalSales desc;"""
cursor.execute(query)
data8 = cursor.fetchall()
df8 = pd.DataFrame(data8, columns = ["PaymentMethod", "TotalPayment"])
df8a = df8.head()
df8a
```

Out[452…

	PaymentMethod	TotalPayment
0	Debit	6778049.0
1	Credit	5124057.0
2	UPI	2587674.0
3	Netbanking	1620846.0
4	COD	843485.0

In [318…

```
# Create a pie chart
plt.figure(figsize=(4, 4))
plt.pie(df8a['TotalPayment'], labels=df8a['PaymentMethod'], autopct='%1.1f%%', s
plt.title('Distribution of Payments by Payment Method')
plt.show()
```



In [272…

```
# Find the products with the highest average time spent on the website by custom
query = """SELECT Product_Name, round(AVG(Time_Spent_on_Website),2) AS AverageTi
FROM dmartready GROUP BY Product_Name
ORDER BY AverageTimeSpent DESC"""
cursor.execute(query)
data9 = cursor.fetchall()
df9 = pd.DataFrame(data9, columns = ["ProductName", "AverageTimeSpent"])
df9.head()
```

Out[272…

	ProductName	AverageTimeSpent
0	Ariel Matic Top Load Liquid Detergent	10.76
1	Amul Taaza Toned Milk	10.58
2	Cavin's Chocolate Milkshake	10.36
3	Tata Sampann Chana Dal	10.30
4	Tata Sampann Moong Dal (Pesara Pappu)	10.26

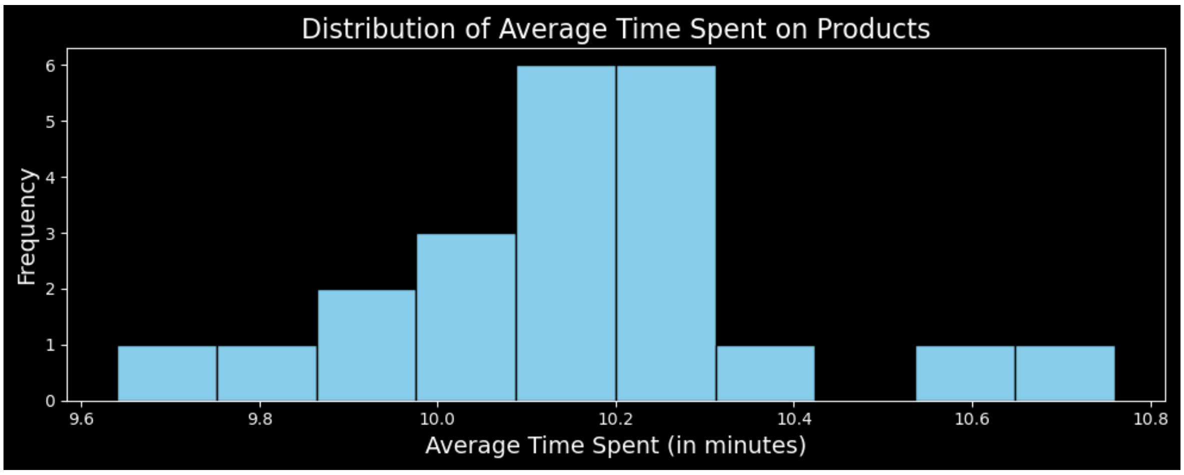
In [340…

```
# Apply dark background style
plt.style.use('dark_background')

# Create a histogram with a dark background
plt.figure(figsize=(10, 4))
plt.hist(df9['AverageTimeSpent'], bins=10, color='skyblue', edgecolor='black')

# Add titles and labels
plt.title('Distribution of Average Time Spent on Products', fontsize=16)
plt.xlabel('Average Time Spent (in minutes)', fontsize=14)
plt.ylabel('Frequency', fontsize=14)

# Show the plot
plt.tight_layout()
plt.show()
```



In [462…

```
# Which states & city have the highest cancellation rates (orders cancelled / to
query = """SELECT State, city,
       (COUNT(Cancellation_Date) / COUNT(Order_ID)) * 100 AS CancellationRate
FROM dmartready GROUP BY State, city
ORDER BY state, CancellationRate DESC;"""
cursor.execute(query)
data10 = cursor.fetchall()
df10 = pd.DataFrame(data10, columns = ["State", "City", "CancellationRate"])
df10.head()
```

Out[462…

	State	City	CancellationRate
0	Andhra Pradesh	Anantapur	8.7302
1	Andhra Pradesh	Tirupati	8.1340
2	Andhra Pradesh	Vijayawada	8.0725
3	Andhra Pradesh	Guntur	7.9646
4	Andhra Pradesh	Rajahmundry	7.5617

In [490…

```
data = {
    'State': ['Andhra Pradesh'] * 10 + ['Gujarat'] * 10 + ['Maharashtra'] * 10 +
    'City': ['Anantapur', 'Tirupati', 'Vijayawada', 'Guntur', 'Rajahmundry', 'Vi
             'Bhavnagar', 'Ahmedabad', 'Rajkot', 'Vadodara', 'Surat', 'Anand', '
             'Kolhapur', 'Thane', 'Amravati', 'Aurangabad', 'Solapur', 'Pune', '
             'Ramagundam', 'Nalgonda', 'Hyderabad', 'Adilabad', 'Nizamabad', 'Kh
    'CancellationRate': [8.7302, 8.1340, 8.0725, 7.9646, 7.5617, 7.1537, 6.8376,
                         7.7922, 7.6800, 7.6792, 7.6583, 7.5353, 6.9307, 6.8910,
                         10.2524, 10.2326, 8.0569, 7.6205, 7.5929, 7.2968, 6.637
                         7.9077, 7.1186, 6.9652, 6.7241, 6.5625, 6.4570, 6.4364,
}

dfdata = pd.DataFrame(data)

# Create the interactive bar plot
fig = px.bar(dfdata,
             x='City',
             y='CancellationRate',
             color='State',
             title='Cancellation Rate by City and State',
             labels={'CancellationRate': 'Cancellation Rate (%)'},
             template="plotly_dark",
             hover_data=['State'])
```

```
fig.update_layout(barmode='group', xaxis_tickangle=-45)
fig.show()
```

In [472…

```
# Total sales in each year ?
query = """select year, sum(Total_Order_Value) as TotalSales from dmartready gro
cursor.execute(query)
data11c = cursor.fetchall()
df11c = pd.DataFrame(data11c, columns = ["Year", "TotalSales"])
df11c.head()
```

Out[472…

	Year	TotalSales
0	2021	5526504.64
1	2022	5729150.56
2	2023	5698454.13

In [496…

```
# Creating a box plot with Plotly Express
fig = px.box(df, x="Year", y="TotalOrderValue", title="Total Sales Distribution
             labels={"TotalSales": "Total Sales", "Year": "Year"},
             template="plotly_dark")

# Show the plot
fig.show()
```

In [512…

```
# Total sales by each year & month ?
query = """select year, month, round(sum(Total_Order_Value)) as TotalSales from
cursor.execute(query)
data11d = cursor.fetchall()
df11d = pd.DataFrame(data11d, columns = ["Year", "Month", "TotalSales"])
# Define the correct month order
month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', '
# Convert the 'Month' column to a categorical type with the specified order
df11d['Month'] = pd.Categorical(df11d['Month'], categories=month_order, ordered=

# Sort the DataFrame based on 'Year' and 'Month'
df_sorted = df11d.sort_values(by=['Year', 'Month'])

# Display the sorted DataFrame
df_sorted.head()
```

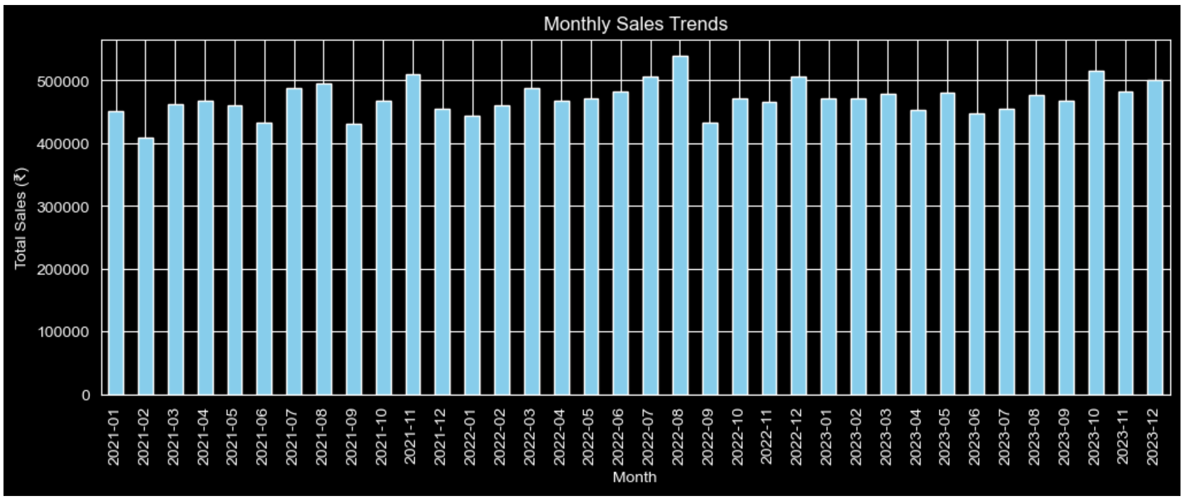
Out[512…

	Year	Month	TotalSales
4	2021	January	450819.0
3	2021	February	408273.0
7	2021	March	462025.0
0	2021	April	466984.0
8	2021	May	461050.0

In [76]:
```
# Plotting the data using Plotly Express
fig = px.line(df_sorted, x="Month", y="TotalSales", color="Year", title="Total S
              labels={"TotalSales": "Total Sales", "Month": "Month", "Year": "Ye
              markers=True)

# Update layout for dark background
fig.update_layout(
    plot_bgcolor='rgb(32, 32, 32)',  # Dark background for the plot area
    paper_bgcolor='rgb(28, 28, 28)',  # Dark background for the entire figure
    font=dict(color='white'),  # Set text color to white for better contrast
    title_font=dict(size=20, color='white'),  # Title font color
    xaxis=dict(showgrid=True, gridcolor='rgb(50, 50, 50)', title_font=dict(color
    yaxis=dict(showgrid=True, gridcolor='rgb(50, 50, 50)', title_font=dict(color
)
# Show the plot
fig.show()
```

In [526…
```
# Sales Trends
plt.style.use('dark_background')
sales_trends = df.groupby(df['OrderDate'].dt.to_period("M"))['TotalOrderValue'].
plt.figure(figsize=(12, 4))
sales_trends.plot(kind='bar', color='skyblue', title='Monthly Sales Trends')
plt.xlabel('Month')
plt.ylabel('Total Sales (₹)')
plt.show()
```



In [222…

```
# Analyze the delivery delay by calculating the difference between the order dat
query = """SELECT Order_ID, DATEDIFF(Delivery_Date, Order_Date) AS DeliveryDelay
FROM dmartready
WHERE Delivery_Date IS NOT NULL"""
cursor.execute(query)
data12 = cursor.fetchall()
df12 = pd.DataFrame(data12, columns = ["OrderID", "DeliveryDelay"])
df12
```

Out[222…

	OrderID	DeliveryDelay
0	479577309	None
1	634865221	None
2	113166210	None
3	740539230	None
4	156544145	None
...	...	...
24995	449449740	None
24996	171612408	None
24997	961751448	None
24998	258727912	None
24999	962056465	None

25000 rows × 2 columns

In [528…

```
# Identify the top 10 products with the highest discount rate (based on the perc
query = """SELECT product_name, (Discount_Price / MRP) * 100 AS DiscountPercenta
FROM dmartready
ORDER BY DiscountPercentage DESC
LIMIT 10"""
cursor.execute(query)
data13 = cursor.fetchall()
df13 = pd.DataFrame(data13, columns = ["ProductName", "DiscountPercentage"])
df13r = df13.round()
df13r
```

Out[528…

	ProductName	DiscountPercentage
0	Nimyle Herbal Anti-Bacterial Floor Cleaner	95.0
1	Nimyle Herbal Anti-Bacterial Floor Cleaner	95.0
2	Usha Heavy Weight Iron - EI3710: 1000 W	95.0
3	Premia Tur Dal Latur (Kandi Pappu)	95.0
4	Tata Tea Chakra Gold Premium Tea	95.0
5	Wagh Bakri Masala Tea	95.0
6	Nutri Delite Breakfast Cereal	95.0
7	Tata Tea Chakra Gold Premium Tea	95.0
8	Tata Tea Chakra Gold Premium Tea	95.0
9	Pigeon Induction Cooktop Acer Plus: 1800 W	95.0

In [468…

```python
# Group by ProductName and calculate the mean DiscountPercentage
df_grouped = df13.groupby('ProductName', as_index=False)['DiscountPercentage'].m

# Create the bar chart
fig = px.bar(df_grouped,
             x='ProductName',
             y='DiscountPercentage',
             title='Discount Percentage by Product',
             labels={'DiscountPercentage': 'Discount Percentage', 'ProductName':
             color='DiscountPercentage',  # Optional: Add color to make the char
             template="plotly_dark",
             text='DiscountPercentage'   # Show DiscountPercentage on top of the
            )

# Rotate x-axis labels for better readability
fig.update_xaxes(tickangle=25)

# Show the plot
fig.show()
```

```
In [234…
# Identify the sales based on gender ?
query = """select gender, round(sum(total_order_value)) as TotalSales from dmart
cursor.execute(query)
data14 = cursor.fetchall()
df14 = pd.DataFrame(data14, columns = ["Gender", "Total"])
df14
```

Out[234…

	Gender	Total
0	Male	9688741.0
1	Female	7265369.0

```
In [240…
# Identify the product wise shipping charges ?
query = """select product_name, sum(shipping_charges) as TotalShippingCharges fr
cursor.execute(query)
data15 = cursor.fetchall()
df15 = pd.DataFrame(data15, columns = ["ProductName", "TotalShippingCharges"])
df15.head()
```

Out[240…

	ProductName	TotalShippingCharges
0	Ariel Matic Top Load Liquid Detergent	42325.0
1	VSR Channa Dal	38875.0
2	Tenali Double Horse Chana Dal	43100.0
3	Tata Tea	43025.0
4	Sofit Soya Drink Vanilla	38475.0

In [538…

```
# Identify the shipmode wise total sales based on order status ?
query = """select ship_mode, order_status, round(sum(total_order_value),2) as To
order_status order by TotalRevenue desc"""
cursor.execute(query)
data16 = cursor.fetchall()
df16 = pd.DataFrame(data16, columns = ["ShipMode", "OrderStatus", "TotalRevenue"
df16.head(6)
```

Out[538…

	ShipMode	OrderStatus	TotalRevenue
0	Free	Delivered	5385100.22
1	Free	Shipped	3753593.82
2	Priority	Delivered	2580485.54
3	Express plus	Delivered	2083202.70
4	Priority	Shipped	1794704.86
5	Express plus	Shipped	1357022.19

In [546…

```
# Marketing Effectiveness
# Identify the total revenue based on (gender & marketing/advertisement) ?
query = """select gender, `Marketing/Advertisement`, sum(Total_Order_Value) as T
group by gender, `Marketing/Advertisement` order by TotalRevenue desc"""
cursor.execute(query)
data17 = cursor.fetchall()
df17 = pd.DataFrame(data17, columns = ["Gender", "Marketing/Advertisement", "Tot
df17.head(10)
```

Out[546…

	Gender	Marketing/Advertisement	TotalRevenue
0	Male	Instagram	3280998.42
1	Female	Instagram	2474980.60
2	Male	Facebook	2430219.62
3	Female	Facebook	1789379.15
4	Male	Other	1582572.71
5	Male	Friends	1565633.57
6	Female	Other	1220959.05
7	Female	Friends	1213493.08
8	Male	TV	829316.24
9	Female	TV	566556.89

In [544…

```
# Create an interactive Sunburst chart
fig = px.sunburst(df17,
                  path=['Gender', 'Marketing/Advertisement'],  # Hierarchical le
                  values='TotalRevenue',  # Values that determine the size of th
                  title="Total Revenue by Gender and Marketing Platform",  # Tit
                  template="plotly_dark",
                  labels={"TotalRevenue": "Total Revenue", "Gender": "Gender", "

# Show the interactive plot
fig.show()
```

In [342…

```
# Profit by each category ?
# Replace "Cancelled" and "Returned" in the 'Order_Status' column with 0
df['TotalOrderValue'] = df['TotalOrderValue'].where(~df['TotalOrderValue'].isin(
```

In [346…

```
df['ShippingCharges'] = df['ShippingCharges'].where(~df['ShippingCharges'].isin(
```

In [352…

```
df['TotalOrderValue'] = pd.to_numeric(df['TotalOrderValue'], errors='coerce')
df['MRP'] = pd.to_numeric(df['MRP'], errors='coerce')
df['DiscountPrice'] = pd.to_numeric(df['DiscountPrice'], errors='coerce')
df['ShippingCharges'] = pd.to_numeric(df['ShippingCharges'], errors='coerce')
```

In [354…

```
df['Profit'] = df['TotalOrderValue'] - (df['MRP'] - df['DiscountPrice'] + df['Sh
```

In [390…

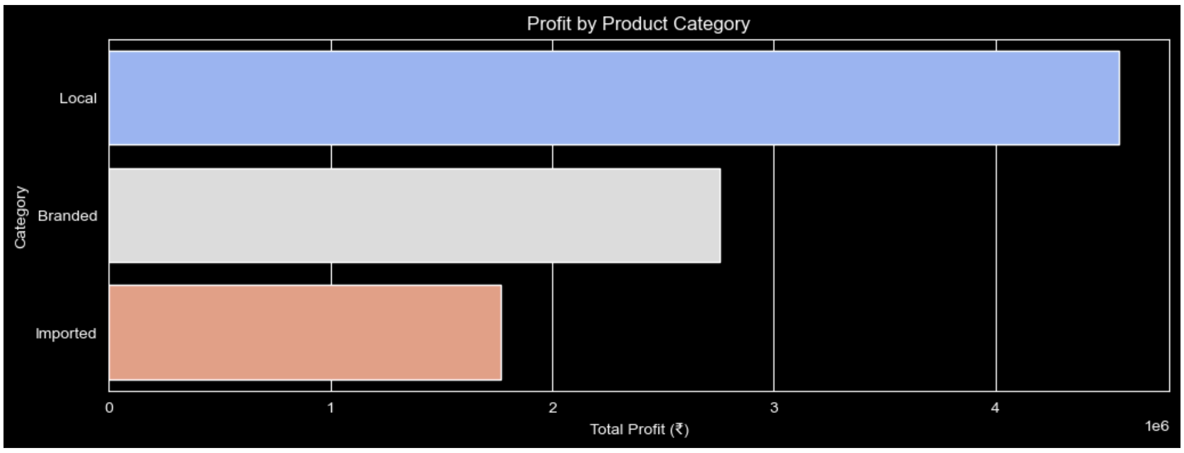
```
profit_by_category = df.groupby('Category')['Profit'].sum().sort_values(ascendin
profit_by_category
```

Out[390…

```
Category
Local       4557246.29
Branded     2754302.54
Imported    1769457.06
Name: Profit, dtype: float64
```

In [392…

```
# Profitability Analysis
plt.style.use('dark_background')
plt.figure(figsize=(12, 4))
sns.barplot(x=profit_by_category.values, y=profit_by_category.index, palette='cc
plt.title('Profit by Product Category')
plt.xlabel('Total Profit (₹)')
plt.ylabel('Category')
plt.show()
```



In [388…

```
summary = {
    'Metric': ['Highest Sales Month', 'Top-Selling Product', 'Best Customer Segm
    'Value': [
        sales_trends.idxmax(),
        product_popularity.idxmax(),
        customer_segmentation.stack().idxmax(),
        avg_delivery_time.idxmin()
    ]
}
summary_df = pd.DataFrame(summary)
print("Summary of Key Insights")
print(summary_df)
```

```
Summary of Key Insights
                  Metric                             Value
0    Highest Sales Month                           2022-08
1    Top-Selling Product  Harpic Bathroom Cleaner - Floral
2  Best Customer Segment                      (Male, 36-45)
3           Fastest City                         Ahmedabad
```

Reporting

Sales Insights

Total Sales: The store generated total sales of ₹16,954,109 over the analyzed period. Average sales per order are ₹678.16, indicating moderate order values.

Yearly Performance: Sales have remained steady across years, with slight peaks in 2022 at ₹5,729,150.56. Orders have grown from 8,310 in 2021 to 8,365 in 2023, suggesting slight growth in demand.

Monthly Trends: November and December consistently show higher sales and orders, likely due to festive shopping. February and September have lower sales and order volumes across years, indicating potential off-seasons.

Category Performance: Local products contribute the most to sales (₹8,482,035) and profit (₹4,557,246.29), emphasizing customer preference for local goods. Branded products are the second-best performer.

Customer Behavior

Gender-Based Insights: Male customers contribute more to sales, especially in the 26-45 age group. Males are the primary audience for Instagram and Facebook campaigns.

State and City Analysis: Maharashtra, Andhra Pradesh, Gujarat, and Telangana are top-performing states, with Maharashtra leading sales at ₹4,298,237. Top cities like Suryapet, Vadodara, and Guntur have significant contributions to overall sales.

Time Spent on the Website: Andhra Pradesh customers spend the most time online (10.50 minutes on average). Cities like Ramagundam, Vijayawada, and Karimnagar have the highest engagement levels.

Product Performance

Top-Selling Products: Household essentials like Harpic Bathroom Cleaner and Parle Nutricrunch Digestive Cookies are top contributors to sales. Products with higher engagement also show high average time spent, such as Ariel Matic Top Load Liquid Detergent.

High Discount Products: Products like Nimyle Herbal Anti-Bacterial Floor Cleaner and Tata Tea Chakra Gold Premium Tea offer the highest discounts (95%), driving sales but potentially affecting margins.

Profitability: Local and branded products contribute significantly to profits, emphasizing a balance between cost and demand.

Operational Insights

Delivery Efficiency: Most deliveries occur within 2-10 days, with a need to improve delivery times for orders taking over a week. Free shipping generates the most revenue, but priority and express plus shipping are growing contributors.

Payment Preferences: Debit cards are the most popular payment method, followed by credit cards. COD (Cash on Delivery) contributes the least, indicating digital payment adoption.

Cancellation Rates: Andhra Pradesh and Gujarat cities like Anantapur and Bhavnagar have high cancellation rates, signaling potential issues with customer satisfaction or delivery logistics.

Marketing Effectiveness

Top Platforms: Instagram campaigns yield the highest revenue (₹5,755,979), especially among male customers. Facebook and referrals from friends also perform well, while TV ads are the least effective.

Gender-Specific Campaigns: Male customers engage more with Instagram, while female customers respond better to Facebook and referral campaigns.

Recommendations

Seasonal Campaigns: Focus on November and December with targeted marketing campaigns and attractive discounts to maximize festive demand.

Improve Delivery Times: Reduce delays for longer delivery periods to improve operational efficiency and customer satisfaction.

Optimize Discounts: Reassess high-discount strategies for products contributing low profitability while promoting profitable items.

City-Specific Strategies: Address high cancellation rates in cities like Anantapur and Bhavnagar by improving customer communication and delivery reliability.

Leverage Top Platforms: Double down on Instagram and Facebook campaigns for male audiences while exploring other platforms for female engagement.

Conclusion

The DmartReady Online Store Analysis highlights significant patterns and insights crucial for strategic decision-making. By leveraging data analysis tools like SQL and Python, the study provides actionable recommendations to enhance sales, improve customer satisfaction, and optimize operational efficiency. The findings underscore the importance of focusing on customer engagement, product diversification, and effective marketing strategies to maintain competitive advantage and drive long-term profitability. Ultimately, this analysis serves as a roadmap for DmartReady to capitalize on its strengths and address areas of improvement, ensuring sustained growth in the competitive e-commerce landscape.

Thank You!

Questions or Feedback? Contact: @shreeram0912

```
In [ ]:
```